# 03-REQUIREMENT ENGINEERING

Definire le proprietà del sistema prima di iniziare il suo sviluppo:

* In maniera molto chiara e precisa

PROPRIETÀ:

* proprietà **funzionali** → cosa il sw deve fare
  + Cosa fa, casi d’uso
  + Dati input 🡪 determinati output
  + Casi d’uso
* Proprietà **NON funzionali** → vincoli che devono essere soddisfatti per fare in modo che il sw faccia ciò che deve fare
  + **Usabilità**
    - Ore per imparare ad usarlo
    - Quanto gli utenti apprezzano il sw
    - Funzioni utili all’utente
  + **Efficienza/Performance**
    - Response time 🡪 da quando schiaccio qualcosa a quando l’app mi risponde
    - Memoria, cpu, banda, energia usata
  + **Correttezza**
    - Capacità di dare il risultato atteso in ogni caso
  + **Affidabilità/Disponibilità**:
    - Difetti visibili all’utente per un periodo di tempo
    - Probabilità di difetti su un periodo di tempo
    - Percentuale di tempo in cui il prodotto non è disponibile per l’utente finale
  + **Mantenibilità**:
    - Quanto effort in personOur per add, modify, cancel di funzioni sw
    - Effort per fixare un difetto
    - Effort per deploy su diverse piattaforme
  + **Portabilità**
    - Effort per risviluppare l’applicazione su un’altra piattaforma
  + **Sicurezza:** autorizzazioni, condivisione, protezione
    - Attacchi che deve essere in grado di evitare
  + **Safety:** assenza di situazioni rischiose per persone
  + **Dependability**: safety + security + reliability

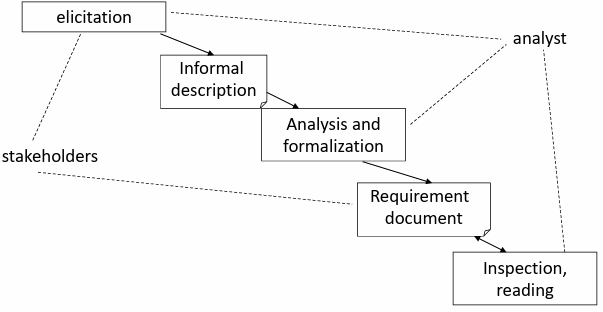
Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

Immagine che contiene testo, schermata, diagramma, Carattere

Descrizione generata automaticamenteNella fase di Development, si inizia con la fase di **Requirement definition**, nella quale si scrive il requirement document.   
Per scriverlo bisogna capire e descrivere cosa fa il sistema

### Come funziona?

* Requirement **elicitation**
  + Parli con utenti
  + Analizzi sistemi simili
  + Brainstorming
  + → ottieni **descrizione informale** (alto livello) del tuo sistema
* **Analizzi** la descrizione e provi a **formalizzarla**
  + → ottieni il **requirement document**
* **Inspection, reading**
  + Verifica del requirement document
    - Deve essere compreso sia dagli esperti di dominio sia da chi non lo è
      * Tutti gli stakeholder devono saper riconoscere i loro bisogni nel documento
      * L’analista effettua un’analisi approfondita del documento

#### Requirements informali

Il punto iniziale è una descrizione informale fornita da un cliente o un utente potenziale.

Se non si definiscono i requisiti correttamente ci possono essere incomprensioni,  
perciò i **requisiti** devono essere:

* **Completi** → deve descrivere le descrizioni di tutte le caratteristiche richieste
* **Consistenti** → non ci devono essere conflitti tra le varie proprietà

*Utopistico avere consistenza e completezza al primo colpo*

**Difetti**:

* Omissioni
* Incorretto
* Inconsistente
* Ambiguo
* Informazioni estranee *→ informazion non utili*
* Ridondanti

#### Tecniche per formalizzare:

* Identificare stakeholder
* Descrivere personas e stories
* Context diagram (ciò che sta dentro e fuori)
* Definire interfacce
* Requisiti
* Scenari e sequence diagrams
* Casi d’uso
* Glossario

##### Identificare stakeholders

Chiedere al cliente una descrizione informale del prodotto che dobbiamo realizzare.

Immagine che contiene testo, schermata, Carattere, documento

Descrizione generata automaticamentePrima di tutto bisogna capire chi è il cliente:

* Utenti che **usano** l’applicazione
  + Possono includere diversi profili
* **Buyer**/commisioner
  + Paga per app
* **Analista**
  + Conosce abbastanza bene il dominio ma anche sw
* **Developer**/sw architect
  + A partire da una descrizione dettagliata, realizza il sw

Gestire tutti gli stakeholder è molto importante per gestire i punti di vista e i correlati requisiti sw

##### STORIES AND PERSONAS

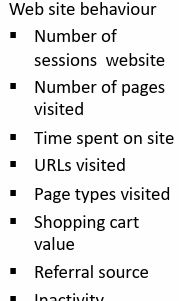
Tecniche informali che definiscono cosa deve fare l’applicazione. Servono per applicativi molto eterogenei che possono usare quasi tutti → mass market.

* Chi sono i clienti che mi porteranno più profitto e cosa vogliono

##### Profilare utente

Immagine che contiene testo, ricevuta, schermata, Carattere

Descrizione generata automaticamenteServe avere delle dimensioni per capire come vendere meglio:

* Demografico
* Psicologico
* Geografico

Inoltre, ci potrebbero essere delle ulteriori informazioni interessanti:

* Prima visita sul sito → do uno sconto
* Visitatore ritornato → ricordo di sconti
* Cliente ritornato → propongo prodotti simili a quelli comprati

##### Context diagram

Definisce cosa è dentro e cosa è fuori dall’app sviluppata:

* **Dentro** → devo sviluppare
* **Fuori** (attore) → necessita interfaccia per poter dialogare con il nostro sistema
  + Immagine che contiene diagramma, schizzo, origami, linea

    Descrizione generata automaticamenteInterfaccia grafica
  + Interfaccia API

Posso farlo **con UML**:

* Rettangolo con **sistema**
* Metto tutte le **interfacce** verso gli attori
  + Omino per umano
  + Generale per sistema

🡪ricavo le interfacce

Immagine che contiene testo, schermata, Carattere, algebra

Descrizione generata automaticamente

* + - Cosa andrò a produrre

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteBisogna realizzare una tabella con:

* Attore
  + Interfaccia fisica
  + Interfaccia logica

**3 tipi di interfacce principali:**

* Interfaccia utente, GUI
* Interfaccia procedurali
* Dati scambiati (codice del prodotto)

*Context diagram importante per avere un accordo con il cliente in modo tale da comprendere l’obbiettivo del sistema e qual è la situazione attuale dalla quale devo partire.*

##### REQUISITI

*Ci sono dei requisiti non funzionali derivati dal dominio.*

Devo dare **un’etichetta** univoca ad ogni requisito ed inserire i requisiti in una forma tabellare

* Immagine che contiene testo, schermata, Carattere, numero

  Descrizione generata automaticamenteRequisiti **funzionali** F: ID+descrizione
  + Dare nomi è utile per comprendere requisiti fatti, testati, rilasciati, pagati, etc.
* **Requisiti non funzionali**
  + Di prodotto
  + Organizzazionali
  + Esterni

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteF1.1 dovrebbe essere un link al requisito funzionale F1.1

Tutti i requisiti devono essere misurabili →

* **Velocità**
  + Immagine che contiene testo, diagramma, schermata, Piano

    Descrizione generata automaticamenteTransizioni processate al secondo
  + Response time
  + Refresh time dello schermo
* **Dimensione**
  + MBytes
  + Nr di chip ROM
* **Facilità di utilizzo**
  + Training time
  + Numero di errori
  + Numero di richieste d’aiuto (ticket)
* **Reliability** 
  + Mean time to failure
  + Probabilità di non disponibilità
  + Disponibilità
* **Robustezza**
  + Tempo per ripartire dopo un fallimento
  + Percentuale di eventi che causano failure
  + Probabilità di corruzione dei dati quando avvengono failure
* **Portabilità**
  + Percentuale di sistemi che riesce a gestire
  + Numero di sistemi target
* Usabilità, efficienza, correttezza, disponibilità, manutenibilità, sicurezza, safety, portabilità

Immagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente

*Conflitti tra requisiti funzionali diversi sono abbastanza comuni per questo bisogna valutare quanto i requisiti non funzionali siano importanti:*

* *Più importante sicurezza o efficienza?*  Domanda a cui devono rispondere a livello business, non dai developer.

**Problemi dei requisiti di dominio:**

* **Undestrandability**
  + Requisiti sono espressi nel linguaggio dell’applicazione di dominio
  + Solitamente non capita dagli ingegneri sw
* **Implicitness**
  + Specialisti di dominio capiscono l’area quindi non credono che serva specificare determinate cose

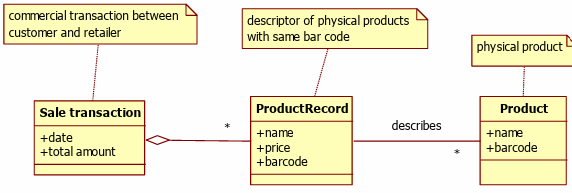
#### Glossario:

Definire il significato dei diversi termini per quanto riguarda il mio progetto

* Esempio:
  + **Vendita**= transazione commerciale tra acquirente e venditore, dove il cliente compra un determinato numero di prodotti dal fornitore
  + Prodotto= -molto più specifico in base a ciò che offro-

#### Class Diagramm

* Descrive application model
* Descrive system design
* Rifinisce il glossario

È un livello abbastanza alto

esiste una saleTransaction con dei ProductRecord, ogni product record ha nome, prezzo, codice a barre.

Un productRecord può descrivere 0+ prodotti

All’interno delle classi:

* Oggetti fisici(aereo, pilota, aeroporto)
* Entità(compagnia, università , dipartimento)
* Descrittori (tipo di aereo)
* Eventi nel tempo (partenza aereo)
* Transazioni commerciali (eventi, scambio di soldi o beni o servizi)
* Intervalli di tempo (volo, viaggio)

## SCENARIO E CASO D’USO

**Scenario**: sequenza di passi che descrivono un’iterazione tipica dell’utente con il sistema.

**Sequenza**: tempo è parte chiave del modello (non di functional, glossary, context diagram, use cases)

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

*Precondition: cassiere non autorizzato*

*Post condition; cassiere autorizzato e autenticato*

*Preconditiion: cassiere autenticato*

**PRECONDITION**: condizione che deve essere verificata prima dell’inizio dello scenario

* Es: bar code è valido; bar code corrisponde ad un prodotto effetivamente disponibile nel negozio

**POSTCODNITION**: condizione soddisfatta alla fine dello scenario

* Es: quantità di denaro a fine transazione= denaro pretransazione +T

#### **Immagine che contiene testo, diagramma, linea Descrizione generata automaticamenteCASI D’USO:**

insieme di scenari con un obiettivo utente comune

Descrivere il comportamento del sistema sotto un insieme di condizioni → come rispondere alle richieste salvaguardando le richieste di tutti gli stakeholders.

* Es: caso d‘uso → gestire vendite
  + S1 → vendere 2 prodotti
  + S2 → vendere 3 prodotti
  + S3 → vendere n prodotti, vendita abortita perché utente non ha denaro
  + S4 → vendere n prodotti, cliente cambia uno dei prodotti

**Elementi chiave:**

* **Attore** 
  + Tipo di utente che interagisce con il sistema
    - Se ho una cassa automatica, il cliente interagisce con il sistema
    - Se ho una cassa classica, il cliente non interagisce con il sistema
* **Sistema** che sarà usato
  + Visto come una black box
    - Parlo di inizio transazione e non di bottone verde premuto
* **Obiettivo** **funzionale** che l’attore ottiene usando il sistema
  + Perché sta usando il sistema?

**Nota su attore:**

* Immagine che contiene testo, schermata, Carattere, linea

  Descrizione generata automaticamenteL’attore è sempre un sottoinsieme degli stakeholder
* L’attore sarà esterno al sistema
* Può essere:
  + Umano
  + Macchine/Sistemi
* Può essere:
  + Primario: inizia iterazione con sistema
  + Secondario: sono passivi

*Es: l’obiettivo del caso d’uso in un bancomat è prelevare i soldi, non inserire il pin*

**Elementi** di un caso d’uso:

* **ACTOR**: qualuno (utente) o qualcosa (Sistema esterno, HW) che:
  + Scambia informazioni con il sistema
  + Fornisce input al sistema o riceve output dal sistema
* Immagine che contiene testo, cerchio, design

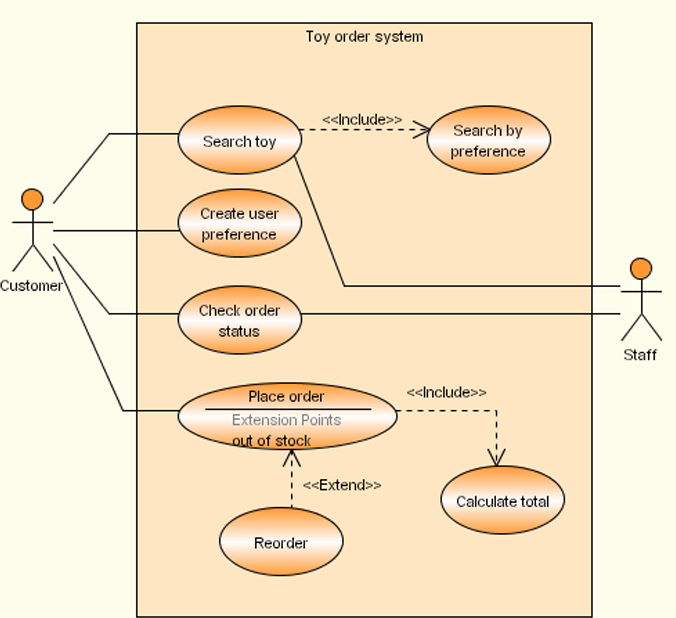
  Descrizione generata automaticamente**Use case**: unità funzionale, parte del sistema
* Immagine che contiene testo, schermata, diagramma, Carattere

  Descrizione generata automaticamente**Metodi** di associazione:
  + Quali attori partecipano al caso d’uso
  + Dove l’esecuzione inizia
  + Molteplicità, direzione…
  + Attore1 partecipa a caso A ed è triggerato dal caso d’uso
  + Attore 2 partecipa a caso d’uso B e use case B è triggerato
* **Include**
  + Immagine che contiene testo, Carattere, schermata, cerchio

    Descrizione generata automaticamentePossiamo dire che la funzionalità A, non esiste senza la funzionalità B
    - Mettere il pin non ha senso se non esiste il login
  + Immagine che contiene testo, schermata, diagramma, design

    Descrizione generata automaticamenteA è una sotto funzionalità di B
* **Generalizzazione**
  + Supervisor ha gli stessi diritti di SalesPerson ma ne ha anche aggiuntive
    - Generico → quello con diritti base
* **Extension**:
  + Immagine che contiene testo, schermata, Carattere, diagramma

    Descrizione generata automaticamenteOpzionale, può avvenire oppure no
  + A può aumentare qualcosa di B
    - Es: in caso di errore



**Requisiti vs casi d’uso**

* Non c’è un mapping 1-1 tra casi d’uso e requisiti
* Ogni volta che creo un caso d’uso devo fare riferimento alla colonna dei requisiti funzionali ed ogni requisito funzionale deve essere almeno in un caso d’uso
* Nota equisiti di alto livello F1, F2 solitamente includono altri casi d’uso; mentre i requisiti di basso livello (F1.1, F1.2) sono inclusi da altri casi d’uso.

Immagine che contiene diagramma, linea, cerchio, testo

Descrizione generata automaticamenteImmagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamenteEsempi:

Immagine che contiene diagramma, testo, cerchio, linea

Descrizione generata automaticamente

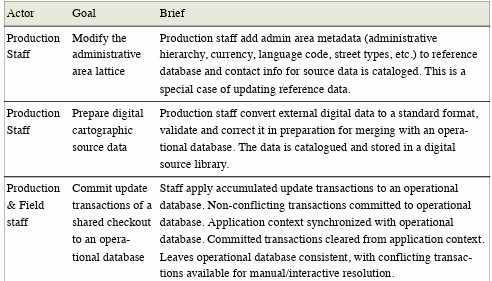
Sappiamo che Check Availability e Acquire Customer data sono obbligatori ma non sappiamo in che ordine avvengo → questo lo definiremo quando parlere mo della narrazione dei casi d’uso

**USE CASE DIAGRAM VS CLASS DIAGRAM**

* Actor 🡪 class
* Use case 🡪 operazione su una classe o più operazioni su più classi
* Interazione → non rappresentare su class diagran

### USE CASE BRIEFS

Serve per definire le attività più importanti → 2/6 frasi per spiegare cosa si fa nel caso d’uso

* + - Qualche informazione in più ma sempre ad alto livello
    - Overview di quello che serve

Attore principale del caso d’uso

Obiettivo che vuole ottenere con quel caso d’uso: perché l’utente utilizza il sistema

Brief → 2/6 frasi

Descrivo il **BestCase/nominal scenario** → descrivo il tutto come se tutto andasse bene.  
**Eccezioni**: quando qualcosa va storto

* **Use case** → titolo caso d’uso
  1. candidatura
* **Scope** → sistema da progettare o porzione del sistema interessato a tale use case (nel caso di sistemi + complessi
  1. Job placement
* **Level** → user-goal (se riguarda iterazione con utente); summary (se racchiudono più casi d’uso)
* **Intention in context** → perché l’utente usa il sistema e cosa vuole ottenere
  1. Il candidato vuole inivare i dati per partecipare ad un colloquio per una posizione lavorativa
* **Primary actor** → utente principale
  1. Candidato ad una posizione lavorativa
* **Support actor** → (se presente) attore di supporto
  1. Non c’è
* **Stakeholders’ interest** → interesse specifico di uno stakeholder (no attori) in questo caso d’uso
  1. C’è
* **Precondition**: situazione (O STATO DEL SISTEMA) che deve essere verificata affichè questo caso d’uso possa avvenire
  1. No precondition (non devo neanche loggarmi)
* **Minimum GUARANTEES**: (se presente) condizione che si verifica dopo l’avvio del caso d’uso (in ogni caso)
  1. No min
* **Success guarantees**: cosa succede quando l’esecuzione di un stato d’uso termina con successo
  1. Dati utente salvati a sistema
* **Trigger**: (se presente) situazione/evento che scatena il caso d’uso da parte del sistema. (nel caso in cui è l’utente ad avviare il caso d’uso autonomamente → no trigger)
  1. No trigger
* **Main success scenario**: elenco puntato in cui c’è un’alternanza (1-1) tra utente e sistema, descrive i passi necessari per raggiungere obiettivo
  1. Utente chiede di effettuare domanda di assunzione
  2. Sistema chiede i dati anagrafici
  3. Utente inserisce dati anagrafici
  4. Sistema salva le informazioni e chiede CV in formato pdf
  5. Utente carica il CV
  6. Sistema salva il CV

→ caso d’uso termina con successo

* **Extension**: valutare tutte le possibili eccezioni che possono capitare nel main success scenario.

1. L’utente annulla: il caso d’uso termina con fallimento
2. Il sistema rileva errori nell’inserimento dati, il sistema mostra un errore ed il caso uso riprende da punto 3
3. uguale a 3a
4. Sistema rileva un errore nel file (integrità o formato), il sistema mostra l’errore e il caso d’uso riprende dal punto 5

#### **DESIGN DEL SISTEMA**

Immagine che contiene diagramma, cerchio, linea, schermata

Descrizione generata automaticamenteEsprime i moduli che compongo il sistema e le loro funzioni di alto livello

Immagine che contiene testo, linea, diagramma, Carattere

Descrizione generata automaticamente→ tutti i sottosistemi (sw+hw) che sono nel context diagram

Nota: il BarCode reader l’ho definito all’interno del sistema → devo metterlo nel design del sistema

Immagine che contiene diagramma, linea, testo, cerchio

Descrizione generata automaticamenteNota2: se non l’avessi messo all’interno ma esterno, non lo avrei dovuto mettere.

Immagine che contiene testo, linea, diagramma, Carattere

Descrizione generata automaticamente

Dunque, il documento dei requisiti deve avere:

* Descrizione generale → text
* Stakeholders → text
* Context diagram e interfacce → UML UCD e Text, PDL,XML, screenshot
* Requisiti → tipo e numerazione
  1. Funzionali
  2. Non funzionali
  3. Dominio
  4. Tabella accessi
* Use case diagram → UML UCD, UC briefs
* Scenarios → Tables, text
* Glossary → Text, UML CD
* Design del sistema → UML CD

### **TECNICHE PER OTTENERE INFORMAZIONI**

* **Focus group**
  + Moderatore + gruppo di persone omogenee
    - Moderatore inizia e monitora i topic della discussione
    - Possono essere aperti o con scaletta
* **Interviste**
  + Domande
* **Questionario**
  + Scritto con risposte aperte/chiuse
    - Persone sono un po’ restie a fornire informazioni con questionari quindi meglio altro
* **Etnografico** 
  + Persona entra in un contesto “nascondendosi” e recupera i bisogni del sistema studiando la situazione
    - Costoso
    - Potrebbe essere molto dispendioso in termini di tempo

**Verifica**:

* Inspection/reading
* Prototyping → sviluppare un piccolo prototipo per i requisiti funzionali
  + Poco tempo per farlo
  + Uso per avviare una discussione con la controparte
  + Molto utile nel caso in cui chi definisce le specifiche non ha una competenza informatica sufficiente per capire
* Iterations
  + Priorità ai requisiti in modo tale da fare uscire versioni intermedie
* MVP → prodotto con funzionalità minime per essere vendute
  + Si testa con gli end user per usare i loro feedback per guidare lo sviluppo successivo
* Model check